ODL Micro-distribution

TWS 16-Dec-19

Tejas Nevrekar, Lumina Networks

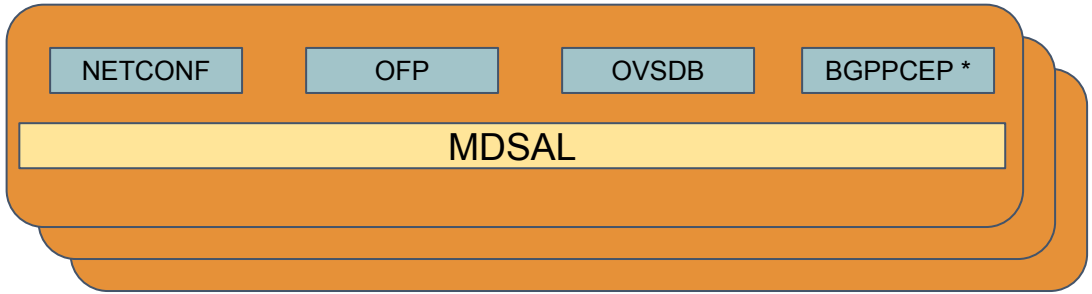
Agenda

- Scope
- ODL Requirements
- Comparison of Karaf alternatives
- Lumina’s ODL-Simple progress & Next steps
- Key implementation steps

Scope

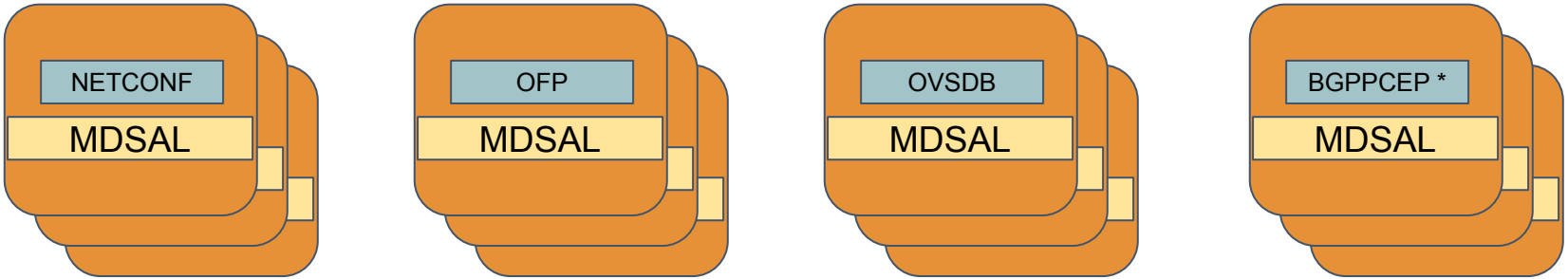
- Plan for adding micro-distribution support in Mg timeframe
- Micro-distribution is making lightweight deployables available for ODL
- Breaking up ODL components into a fine-grained set of micro-services not necessary

What is Micro-distribution of ODL



All in One ODL

Micro-dist based



ODL Requirements

- Statically Inject Service classes from same module(s)
- Statically Inject Service classes from different module(s)
- Read static configuration at start-up
- Reconfigure on dynamic configuration changes
- Create and register a new Service on the fly
- Lookup Service using identifier/keyword
- Allow CLI to manage configuration, logging and functional commands
- Build time definition of distribution, karaf like feature not really needed

Alternatives (and ODL only needs 1)

- Spring (last commit 3 days ago)
  - XML driven configuration
  - Compile+Runtime framework
  - DI (Dependency Injection) and a large ecosystem
- Guice (last commit 16 days ago)
  - Annotation/Code first approach
  - Compile+Runtime framework
  - Focussed only on DI
- Dagger (last commit 6 days ago)
  - Compile time only framework
  - Looks to have advantages over Guice and better maintained
  - https://docs.google.com/presentation/d/1fby5VeGU9CN8zjw4lAb2QPPsKRxx6mSwCe9q7ECNSJQ/pub?start=false&loop=false&delayms=3000&slide=id.p

ODL-Simple

- Statically Inject Service classes from same module(s)
- Statically Inject Service classes from different module(s)
- Read static configuration at start-up
- Reconfigure on dynamic configuration changes
- Register a dynamic Service on the fly
- Lookup dynamic service using identifier/keyword
- Allow CLI to manage configuration, logging and functional commands

Supported Not Supported Partially Supported

Next steps

- Gerrit for removing dependency on mycila
  - https://git.opendaylight.org/gerrit/c/infrautils/+/86037
- Gerrit for different projects - controller, aaa, infrautils, openflowplugin, ovsdb, netconf OR add in distribution
  - https://github.com/tnevrekar/opendaylight-simple/tree/lumina-netconf

Key Implementation Steps

1. Replace all exposed blueprint XMLs with google.guice.AutowiringModule subclasses

2. Install required dependent modules

3. Expose required services using annotation for binding by odl:type

4. Refer required services using the annotation

```
public class NetconfModule extends AutoWiringModule {


    @Override
    protected void configureMore() {
        LOG.info("Loading netconf");
        // Guice
        install(new AnnotationsModule());
        // Controller/MD-SAL
        install(new InMemoryControllerModule());
```

```
@Provides
@Singleton
@GlobalWorkerGroup
EventLoopGroup getGlobalWorkerGroup() {
    return NioEventLoopGroupCloseable.newInstance(0);
}
```

```
@Provides
@Singleton
@org.opendaylight.netconf.simple.NetconfClientDispatcher
NetconfClientDispatcher getNetconfClientDispatcher(
        @GlobalBossGroup EventLoopGroup globalBossGroup,
        @GlobalWorkerGroup EventLoopGroup globalWorkerGroup,
        @GlobalTimer Timer globalTimer) {
    return new NetconfClientDispatcherImpl(globalBossGroup, g
}
```

Key Implementation Steps

5. Add AutoWiring class for reading configuration or initializing

```
@Singleton
public class OpenFlowJavaWiring {
```

6. Read from config using ConfigReader

```
@Inject
public OpenFlowJavaWiring(ConfigReader configReader,
        SwitchConnectionProviderFactory switchConnectionProviderFactory)
    SwitchConnectionConfig defaultSwitchConnConfig = configReader
            .read("/initial/default-openflow-connection-config", SwitchC
                    "openflow-switch-connection-provider-default-impl");
    SwitchConnectionProvider defaultSwitchConnProvider = switchConnectio
            .newInstance(defaultSwitchConnConfig);
```

7. Use Wiring in the Module

```
@Provides
@Singleton SwitchConnectionProviderList getOpenFlowJavaWiring(OpenFlowJavaWi
    return openFlowJavaWiring.getSwitchConnectionProviderList();
}
```

Key Implementation Steps

8. Alternatively move most of the blueprint definition to Annotations in the Code in the respective project. E.g. as done in OVSDB - https://git.opendaylight.org/gerrit/c/ovsdb/+/79782

```
@Singleton
public class HwvtepSouthboundProvider implements Clus
    @Inject
    public HwvtepSouthboundProvider(@Reference final DataBroker
```

9. No additional explicit wiring needed in simple if blueprint.xml replaced by annotations.

```
public class OvsdbModule extends AutoWiringModule {

    public OvsdbModule(GuiceClassPathBinder classPathBinder) {
        super(classPathBinder, "org.opendaylight.ovsdb");
    }

}
```

ODL Simple Runtime statistics

	Clean Start Time	Clean Init Time	Clean Time duration	Next Start Time	Next Init Time	Next Time Duration
ODL-Neon-SR3	0:07:13	0:08:47	0:01:34	0:16:39	0:17:10	0:00:31
	0:12:35	0:13:19	0:00:44	0:17:36	0:18:03	0:00:27
	0:13:52	0:14:19	0:00:27	0:18:26	0:18:54	0:00:28
	0:14:57	0:15:41	0:00:44	0:19:34	0:20:03	0:00:29
Average			**0:00:52**			**0:00:29**

	Start Time	Init Time	Net Time Duration
ODl-Simple-Neon-GA	1:44:16	1:44:28	0:00:12
	1:46:02	1:46:15	0:00:13
	1:46:35	1:46:46	0:00:11
	1:47:29	1:47:40	0:00:11
Average			**0:00:12**

Thanks